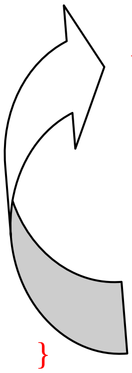
Single thread Scheduler

All processes called once each sample



```
void main(void) {

        init_routines();

        done = 0;
        while (!done) {

                perform_process1(); // Highest priority process
                perform_process2();
                perform_process3();// Lowest priority

                wait_for_sample_period();  //waste time here waiting for sample period to expire
        }
}
```

What if the three processes have different sample rates? With a single thread scheduler the following would work but all sample rates would have to be a multiple of the fastest rate.

```
void main(void) {
        init_routines();
        done = 0;
        samplecnt = 0
        while (!done) {

                perform_process1(); // Call every sample

                if ((samplecnt%5)==0) perform_process2();  // Call every 5 samples

                if ((samplecnt%12)==0) perform_process3(); // Call every 12 samples

                wait_for_sample_period();  //waste time here waiting for sample period to expire
                samplecnt = samplecnt+1;
        }
}
```

Here the problem becomes that all processes (in this case all three) need to complete in the time of one sample period. For example at samplecnt = 60 all three processes are called during a single sample period. What if process 3 takes longer than a single sample rate?

Hardware Interrupt Scheduler

Interrupt Source Vector Table in DSP's Memory

……………	…………..
Serial Port (INT10)	Serial_ISR()
……………	…………..
Timer 0 (INT14)	Timer0_isr()
Timer 1 (INT15)	Timer1_isr()

```
int samplecnt = 0 // Global integer

void main(void) {

    init_routines();

    done = 0;
    while (!done) {

        if ((samplecnt%12)==0) {
            //Call every 12 samples
            perform_process3();
        }

    }

}
```

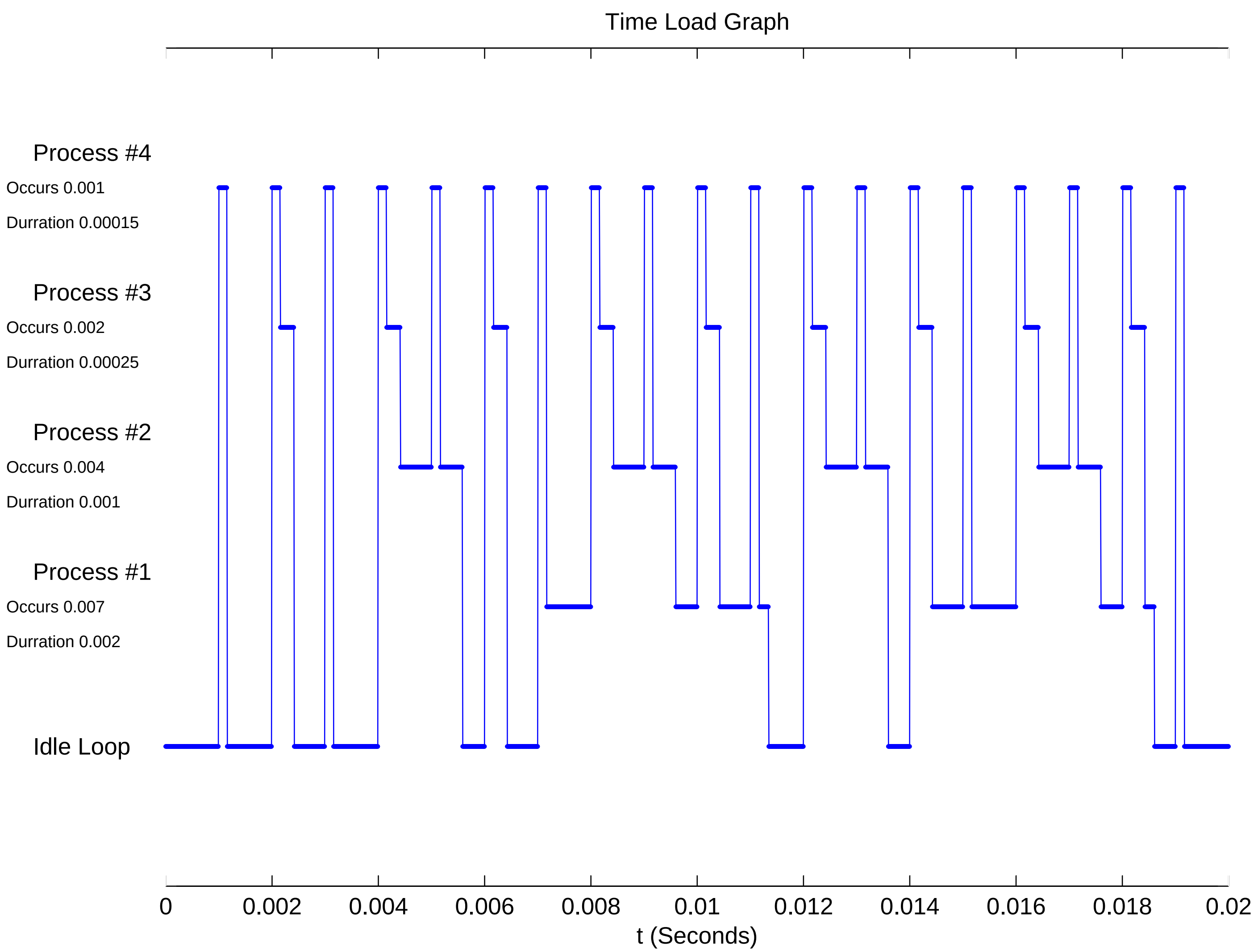
```
// timer 0 interrupts at Sample Period
void timer0_isr(void) {

    perform_process1();

    samplecnt = samplecnt + 1;

}
```

Could timer0 interrupt timer1 or vice versa? Yes the DSP could be setup to do that but in many cases (and in our case) it is better practice to not allow this to happen. You have to be careful with certain assembly instructions when writing ISR routines to make them "interruptable".

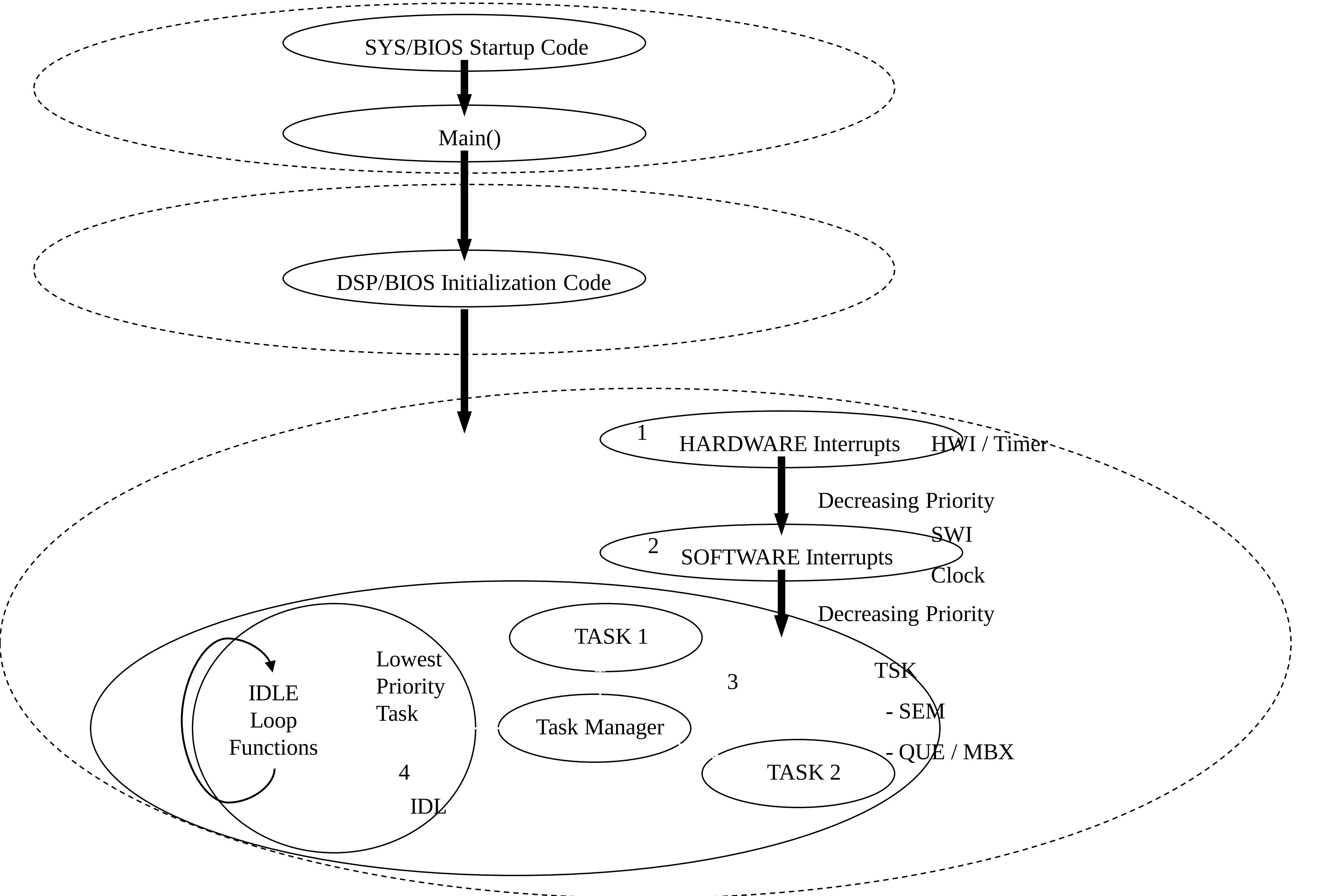
```
// timer 1 interrupts at 5 X Sample Period
void timer1_isr(void) {

    perform_process2();

}
```

The while loop inside main() now becomes the low priority processing loop. Also called the "background" loop. Process 1 and Process 2 have the highest priority. When either timer 0 or timer 1 counts down to 0, the DSP's hardware automatically stops the current code running in the background loop and jumps to the function specified in the Interrupt Vector Table, in this case timer0_isr() or timer1_isr(). When the DSP is done running the instructions in the corresponding interrupt service routine (ISR), the DSP's hardware automatically returns and continues processing where it left off in the background loop.

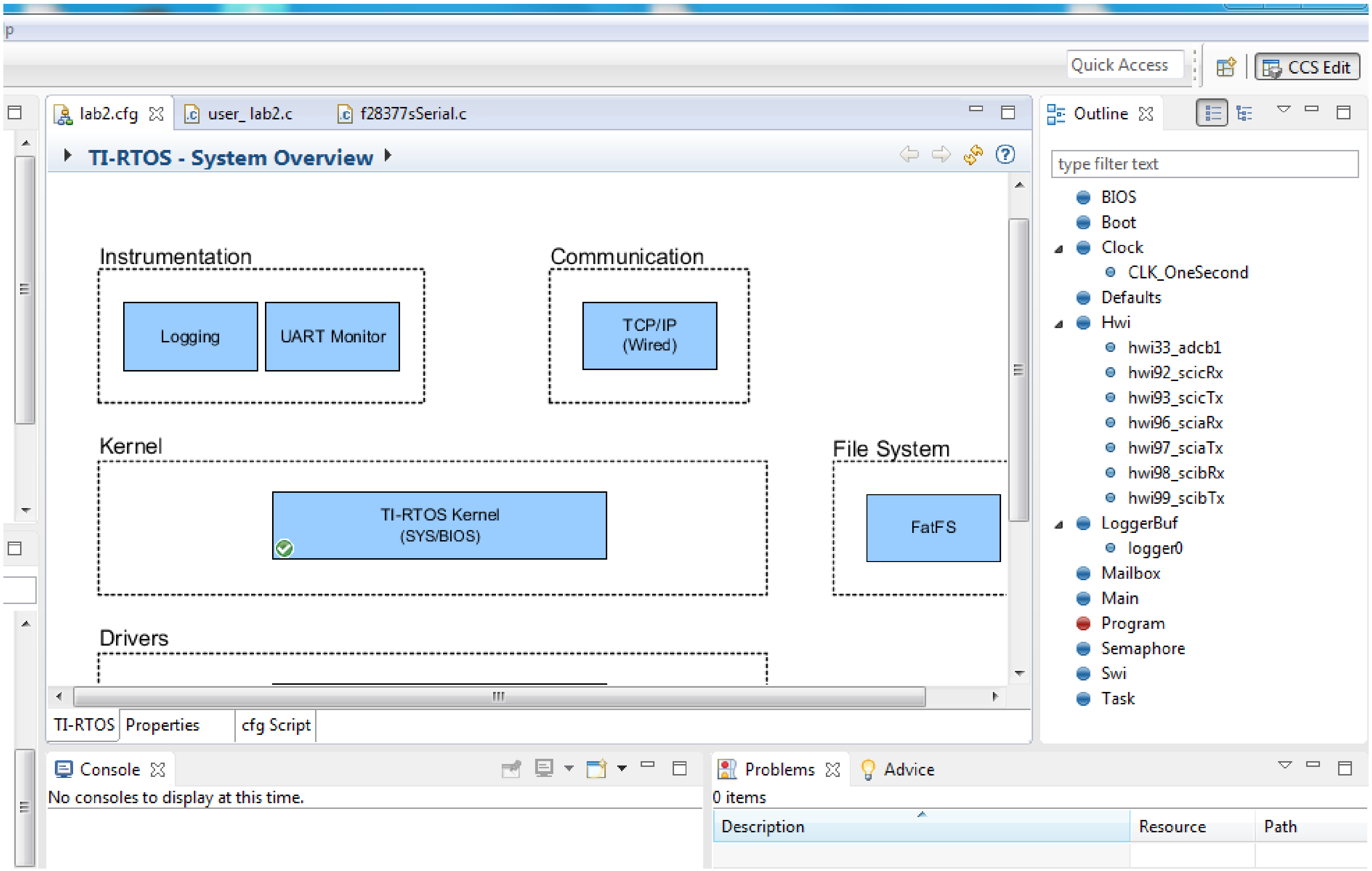
If timer 0 and timer 1 timeout at exactly the same time, timer 0 has the highest priority so its code will run first to completion and then timer 1's code will be executed. If a timer interrupt occurs while the other timer's interrupt service routine code is running, the running code continues to completion and then the other timer's code is executed.


Time Load Graph
Process #4
Occurs 0.001
Durration 0.00015
Process #3
Occurs 0.002
Durration 0.00025
Process #2
Occurs 0.004
Durration 0.001
Process #1
Occurs 0.007
Durration 0.002
Idle Loop
0
0.002
0.004
0.006
0.008
0.01
0.012
0.014
0.016
0.018
0.02
t (Seconds)

SYS/BIOS



SYS/BIOS Startup Code
Main()
DSP/BIOS Initialization Code
1
HARDWARE Interrupts
HWI / Timer
Decreasing Priority
SWI
2
SOFTWARE Interrupts
Clock
Decreasing Priority
TASK 1
3
TSK
- SEM
- QUE / MBX
Lowest
Priority
Task
IDLE
Loop
Functions
Task Manager
TASK 2
4
IDL


Quick Access
CCS Edit
lab2.cfg
user_lab2.c
f28377sSerial.c
TI-RTOS - System Overview
Instrumentation
Logging
UART Monitor
Communication
TCP/IP
(Wired)
Kernel
TI-RTOS Kernel
(SYS/BIOS)
File System
FatFS
Drivers
TI-RTOS
Properties
cfg Script
Outline
type filter text
BIOS
Boot
Clock
CLK_OneSecond
Defaults
Hwi
hwi33_adcb1
hwi92_scicRx
hwi93_scicTx
hwi96_sciaRx
hwi97_sciaTx
hwi98_scibRx
hwi99_scibTx
LoggerBuf
logger0
Mailbox
Main
Program
Semaphore
Swi
Task
Console
No consoles to display at this time.
Problems
Advice
0 items
Description
Resource
Path

SYS/BIOS Example